ONR

Principal Investigator(s) Name: C. J. Walter, M. C. McElvany
PI Institution: Aerospace Technology Center, Allied-Signal Aerospace Company
PI Phone Number: (301) 964-4082, (301) 964-4158
PI E-mail Address: chris@batc.allied.com, michelle@batc.allied.com
Contract Title: Reliability Modeling of Dependable Distributed Systems
Contract Number: N00014-91-C-0014
Reporting Period: 91–May–01 to 91–Jul–31

1 Numerical Productivity Measures

1. Michelle McElvany gave a talk at the Allied-Signal Digital Systems Technology Exchange Conference in June, 1991, on the potential impact of VHDL on dependable distributed system design, analysis and validation.
2. Chris Walter chaired a session and served on the program committee at FTCS in June, 1991, and will do so for FTCS in June, 1992.
3. Chris Walter presented an invited paper at the SEI Fault Tolerant Systems Practitioner's Workshop in June, 1991.

Principal Investigator(s) Name: C. J. Walter, M. C. McElvany
PI Institution: Aerospace Technology Center, Allied-Signal Aerospace Company
PI Phone Number: (301) 964-4082, (301) 964-4158
PI E-mail Address: chris@batc.allied.com, michelle@batc.allied.com
Contract Title: Reliability Modeling of Dependable Distributed Systems
Contract Number: N00014-91-C-0014
Reporting Period: 91–May–01 to 91–Jul–31

2 Detailed Summary of Technical Progress

The overall goal of this project is to develop theory for improved reliability modeling of systems with mixed fault types. This also provides the basis for formal methods to be used in the specification, design, construction and verification of ultra-reliable multi-processor systems. We assume that a *fault* is an anomalous physical condition, the identified or hypothesized cause of an *error*, which may eventually lead to a *failure*, a loss of service.

In the initial project phase, our goal is to develop a hybrid fault and static reliability model that addresses mixed fault types. An in depth study of faults, including their sources, their manifestations, and the techniques needed to reduce malicious fault effects, will then provide accurate inputs to the hybrid model. We partitioned our work into three tasks: analysis of static reliability models, investigation of faults, and evaluation of the impact of fault containment on fault effects and on system reliability.

Since existing fault taxonomies did not adequately address the errors caused by faults, we developed the *hybrid fault taxonomy*. We applied this taxonomy to existing static reliability models and showed them to be either unrealistic in their fault assumptions or too restrictive in the algorithms required to tolerate faults. As a result of this analysis, we developed the ***hybrid fault model*** which supports more realistic fault assumptions and fault tolerance techniques. We also evaluated the hybrid fault model as a systems analysis tool, indicating directions for further research. Details of our progress are reported below.

2.1 Hybrid Fault Taxonomy

Based on our survey of fault taxonomies, described in [1], we have developed the ***hybrid fault taxonomy***, which describe faults based on characteristics of the errors they cause and on the techniques needed to tolerate them. The following definitions, including the fault attributes of *malice* and *symmetry*[1], are essential to the hybrid fault taxonomy, formally defined in §2.1.2.

2.1.1 Terminology

The *scope* of a fault refers to the portion of the system affected by that fault, also called the *fault extent*. A system is *fault tolerant* or *tolerates a fault* if the required system services are maintained in the presence of a fault. Fault *coverage* is a measure of the system's ability to operate correctly

[1]The definitions of malice and symmetry used in [1] are equivalent to the more formal definitions appearing in this report.

in the presence of a particular fault set. *Perfect coverage* assumes that system fault toleration mechanisms are successful for all possible fault sets.

Active or dynamic redundancy attempts to achieve fault-tolerance by fault-detection alone, or in conjunction with location and recovery. Active redundancy techniques include hardware redundancy, using duplication with comparison, standby sparing, or a pair and a spare; information redundancy, using data encoding, parity checks, range checks, or sanity checks; and time redundancy, using checkpointing and rollbacks. [2] In static reliability modeling, all faults are assumed to be permanent; so, no fault location or recovery is considered, and hardware sparing is not used.

Passive or static redundancy uses fault masking to hide the occurrences of faults and to eliminate the effects of the faults, thus avoiding *errors.* In their simplest form, these techniques make no attempt to detect the fault, much less its source, and are transparent to the user or operator. Hardware redundancy, using $3,4$ or N hardware modules simultaneously, is another commonly used passive technique. For further details, see [2].

Passive redundancy techniques can be *iterative* or *non-iterative. Iterative* fault masking techniques, such as interactive convergence [3, 4] and interactive consistency [4, 5], require multiple rounds or iterations of message exchange among participants. *Non-iterative* passive redundancy techniques require a single round of message exchange. Fault-tolerant voting techniques, such as majority and median, are non-iterative passive redundancy techniques on which iterative passive redundancy techniques are often based.

Hybrid redundancy combines active and passive redundancy techniques to support masking and detection of faults. Using hybrid redundancy is usually more expensive and complex than using separate techniques.

All faults are either *non-malicious* or *malicious.* A *non-malicious fault* can be detected using an active redundancy technique. A *malicious fault* cannot be detected using active redundancy techniques, but requires masking using a passive redundancy technique. It should be noted that passive redundancy techniques are capable of tolerating non-malicious faults as well. The difference between non-malicious and malicious faults lies in the *requirement* that passive redundancy techniques be used for malicious faults. Malicious faults are more severe than non-malicious faults.

Timing faults [6, 7], omission faults [6] and crash faults [8] are all examples of non-malicious faults. Faults which alter the contents of a message in a detectable way, such as by violating a range check or a parity check, are also non-malicious. Byzantine faults [9] are malicious.

Faults can be either *symmetric* or *asymmetric.* A *symmetric fault* generates errors that are manifested identically throughout the scope of the fault, *i.e.*, the portion of the system or component affected by the fault. An *asymmetric fault* generates errors that are manifested differently throughout the scope of the fault. Asymmetric faults are more severe than symmetric faults.

2.1.2 Hybrid Fault Classes

Based on the previous definitions, if $\mathcal{F}$ denotes all possible faults, $\mathcal{B}$ all non-malicious faults, $\mathcal{M}$ all malicious faults, $\mathcal{S}$ all symmetric faults, and $\mathcal{A}$ all asymmetric faults, then

$$\mathcal{F} = \mathcal{B} \bigcup \mathcal{M} = \mathcal{A} \bigcup \mathcal{S},$$
$$\text{with} \quad \mathcal{B} \bigcap \mathcal{M} = \emptyset, \text{ and } \mathcal{A} \bigcap \mathcal{S} = \emptyset.$$

We combined the attributes of malice and symmetry to produce the four fault sets that make up the hybrid taxonomy. The set of ***non-malicious symmetric*** faults, $\mathcal{B}_\mathcal{S}$, is given by $\mathcal{B}_\mathcal{S} \equiv \mathcal{B} \bigcap \mathcal{S}$.

The set of *non-malicious asymmetric* faults, $\mathcal{B}_{\mathcal{A}}$, is given by $\mathcal{B}_{\mathcal{A}} \equiv \mathcal{B} \bigcap \mathcal{A}$. The set of *malicious symmetric* faults, $\mathcal{M}_{\mathcal{S}}$, is given by $\mathcal{M}_{\mathcal{S}} \equiv \mathcal{M} \bigcap \mathcal{S}$. The set of *malicious asymmetric* faults, $\mathcal{M}_{\mathcal{A}}$, is given by $\mathcal{M}_{\mathcal{A}} \equiv \mathcal{M} \bigcap \mathcal{A}$.

Definition 1 (Hybrid Fault Taxonomy)
Every fault in $\mathcal{F}$ is in exactly one of the sets $\mathcal{B}_{\mathcal{S}}$, $\mathcal{B}_{\mathcal{A}}$, $\mathcal{M}_{\mathcal{S}}$, or $\mathcal{M}_{\mathcal{A}}$, with

$$\mathcal{F} \equiv \mathcal{B}_{\mathcal{S}} \bigcup \mathcal{B}_{\mathcal{A}} \bigcup \mathcal{M}_{\mathcal{S}} \bigcup \mathcal{M}_{\mathcal{A}}.$$

The definitions of malice and symmetry guarantee that that the individual sets $\mathcal{B}_{\mathcal{S}}$, $\mathcal{B}_{\mathcal{A}}$, $\mathcal{M}_{\mathcal{S}}$, and $\mathcal{M}_{\mathcal{A}}$, are pairwise disjoint. The *worst-case*, or most severe, faults in $\mathcal{F}$ are those in $\mathcal{M}_{\mathcal{A}}$. Faults in $\mathcal{M}_{\mathcal{S}}$ are less severe than the faults in $\mathcal{M}_{\mathcal{A}}$, but are more severe than faults in $\mathcal{B}$.

2.2 Static Reliability Models

We used the taxonomy of §2.1.2 to classify common static reliability models, where all faults are assumed to be permanent, no fault isolation or repair is attempted, and fault coverage is perfect. The reliability of each model is indicated along with the number of faults tolerated by a given system under the assumption of identical nodes. For simplicity, we assumed a synchronous message passing system of m components or processes, called *nodes*, where the only evidence of a faulty node is an error in a message from that node. A good node is expected to collect information from other nodes and to arrive at a local decision that is consistent with decisions of all other good nodes.[2] A good node may also need to compute a local value within a prespecified range of the values of other good nodes. If we assume that all the nodes are identical with reliability R, then the reliability of a system requiring a minimum of n non-faulty nodes to provide system services is given in [10] by

$$R(n \text{ } of \text{ } m) = \sum_{j=n}^{m} \binom{m}{j} R^j (1-R)^{m-j}. \tag{1}$$

Except for the unified model, all of the models presented below assume that every fault is potentially the worst-case fault.

2.2.1 Non-malicious faults ($\mathcal{F}_{\mathcal{B}}$)

All faults are in $\mathcal{F}_{\mathcal{B}}$, where $\mathcal{F}_{\mathcal{B}} \equiv \mathcal{B} \equiv \mathcal{B}_{\mathcal{S}} \bigcup \mathcal{B}_{\mathcal{A}}$ and can be detected by active redundancy techniques. Since all faults that are assumed to occur can be recognized as such, $m_{\mathcal{B}} = f_{\mathcal{B}} + 1$ nodes are needed to guarantee correct operation in the presence of $f_{\mathcal{B}}$ faults. The reliability of this model is given by Equation 1 as $R(1 \text{ } of \text{ } m_{\mathcal{B}})$. This model is overly optimistic, as it assumes that malicious faults can't occur.

2.2.2 Symmetric faults ($\mathcal{F}_{\mathcal{S}}$)

All faults are assumed to be symmetric, and potentially malicious, where $\mathcal{F}_{\mathcal{S}} \equiv \mathcal{S} \equiv \mathcal{B}_{\mathcal{S}} \bigcup \mathcal{M}_{\mathcal{S}}$. By definition, passive redundancy techniques are required to mask such faults. However, since all

[2]If only one of many nodes is good, it may be impossible to identify the correct node. Such identification is beyond the scope of this report. However, we do guarantee that the good node always holds a correct value or a value consistent with all other good nodes' values.

the faults are symmetric, non-iterative passive redundancy techniques, such as majority or fault-tolerant voting, are sufficient to mask any faults. Since a majority of good values is required to compute an accurate value in the presence of faults, a minimum of m_S nodes is required to tolerate f_S malicious symmetric faults, where $m_S = 2f_S + 1$. A minimum of n_S non-faulty nodes is required to keep the system operational, with $n_S = m_S - \lfloor\frac{m_S-1}{2}\rfloor$, where $\lfloor x\rfloor$ is the largest integer not greater than x. The reliability of the system is given by Equation 1 as $R(n_S \ of \ m_S)$, with a minimum of three nodes required to tolerate a single symmetric malicious fault. This model is also overly optimistic, as it does not handle asymmetric non-malicious faults, and assumes that Byzantine or asymmetric malicious faults can't occur.

2.2.3 Asymmetric faults ($\mathcal{F}_{\mathcal{A}}$)

When all faults are asymmetric, and potentially malicious, with $\mathcal{F}_{\mathcal{A}} \equiv \mathcal{A} \equiv \mathcal{B}_{\mathcal{A}} \bigcup \mathcal{M}_{\mathcal{A}}$, the non-iterative passive redundancy techniques described earlier are no longer adequate. Instead, iterative algorithms such as interactive convergence [3, 4] and interactive consistency [4]-[5], with multiple rounds of message exchange, are required to mask these faults. A minimum of $m_{\mathcal{A}}$ processors is required to tolerate $f_{\mathcal{A}}$ faults in $\mathcal{F}_{\mathcal{A}}$, with $m_{\mathcal{A}} = 3f_{\mathcal{A}} + 1$. So, under this model, a minimum of four nodes is needed to tolerate a single, potentially malicious, asymmetric fault. From Equation 1, the reliability is given by $R(n_{\mathcal{A}} \ of \ m_{\mathcal{A}})$ with $n_{\mathcal{A}} = m_{\mathcal{A}} - \lfloor\frac{m_{\mathcal{A}}-1}{3}\rfloor$. This model is overly pessimistic, because not all faults are malicious asymmetric. Furthermore, the extra rounds of message exchange increase the complexity and cost of the algorithms needed to tolerate the faults assumed by the model.

2.2.4 Mixed faults—the unified model ($\mathcal{F}_{\mathcal{U}}$)

Unlike the previous models, the unified model [11] is not limited to a single fault type. Instead, the three fault types used in previous models are supported in a single model. Using the unified model, we have $\mathcal{F}_{\mathcal{U}} \equiv \mathcal{F}_{\mathcal{B}} \bigcup \mathcal{F}_{\mathcal{S}} \bigcup \mathcal{F}_{\mathcal{A}}$, giving $\mathcal{F}_{\mathcal{U}} \equiv \mathcal{F}$. The unified model maximizes the number of non-malicious and symmetric malicious faults which can be tolerated, under the constraint of an upper bound on the number of malicious asymmetric faults.

The $Z(r)$ algorithm, similar to the Oral Messengers ($OM(q)$) algorithm of [12] used for up to q arbitrarily malicious faults, achieves interactive consistency in the presence of $f_{\mathcal{U}} = 2f_{\mathcal{A}} + 2f_{\mathcal{S}} + f_{\mathcal{B}}$ faults, when there are at least $m_{\mathcal{U}}$ processors, with $m_{\mathcal{U}} = f_{\mathcal{U}} + r + 1$, and $r \geq f_{\mathcal{A}}$ rebroadcast rounds. Because the assumption of multiple fault types requires the use of more than just the simple *R(n of m)* from Equation 1, the reliability formula is more complex, and appears in [13].

This model shows improvement over the previous model for numbers of nodes, $m_{\mathcal{U}}$, which cannot be written as $3k + 1$ for any integer k. For example, consider both the $OM(1)$ and $Z(1)$ algorithms with five nodes ($m_{\mathcal{U}} = 3k + 2$, where $k = 1$). Since the $OM(1)$ algorithm treats all other faults as if they were in $\mathcal{F}_{\mathcal{A}}$, only one fault of any type is tolerated. In contrast, with five nodes, the algorithm $Z(1)$ tolerates a single fault in $\mathcal{F}_{\mathcal{A}}$ and a single fault in $\mathcal{F}_{\mathcal{B}}$; or, three faults in $\mathcal{F}_{\mathcal{B}}$; or, one fault in $\mathcal{F}_{\mathcal{S}}$ and one in $\mathcal{F}_{\mathcal{B}}$. Unfortunately, this model applies only to interactive consistency algorithms using the $Z(r)$ algorithm with r rounds of rebroadcast.

2.3 The Hybrid Fault Model

We developed the ***hybrid fault model***, which recognizes mixed fault types, by combining the fault models described in §2.2. The fault tolerance algorithms used in conjunction with the models in §2.2 are replaced by hybrid algorithms that assume mixed faults, as defined in §2.4 and §2.5.

The mixed fault sets used in the hybrid model are combinations of the sets $\mathcal{B}_S$, $\mathcal{B}_A$, $\mathcal{M}_S$, and $\mathcal{M}_A$ (listed in increasing order of severity) that were defined in § 2.1.2. The type of hybrid redundancy algorithm, active or passive, required to tolerate all faults in the assumed fault set is indicated. The type of passive hybrid redundancy algorithm required, iterative or non-iterative, is a function of the worst-case fault type, the data type, and the application. When the fault set requires iterative redundancy, the choice of a hybrid interactive consistency or a hybrid interactive convergence algorithm also depends on the application and data type. If no hybrid algorithms are used, then the hybrid model reverts to a combination of the non-malicious, symmetric, and asymmetric models described in §2.2, with no improvement. Thus, associating the proper hybrid algorithm with the assumed system or node fault set is the key to the hybrid model, as defined below.

Definition 2 (Hybrid Fault Model)

The ***hybrid fault model*** consists of three fault scenarios $H_{\mathcal{B}}$, $H_{\mathcal{MS}}$, $H_{\mathcal{MA}}$, determined by the faults assumed to be tolerated. The corresponding fault sets are given by $\mathcal{F}_{\mathcal{B}}$, $\mathcal{F}_{\mathcal{MS}}$, and $\mathcal{F}_{\mathcal{MA}}$, with

$$\begin{aligned} &\mathcal{F}_{\mathcal{B}} \equiv \mathcal{B}_S \bigcup \mathcal{B}_A, \\ &\mathcal{F}_{\mathcal{MS}} \equiv \mathcal{B}_S \bigcup \mathcal{B}_A \bigcup \mathcal{M}_S, \text{ and} \\ &\mathcal{F}_{\mathcal{MA}} \equiv \mathcal{B}_S \bigcup \mathcal{B}_A \bigcup \mathcal{M}_S \bigcup \mathcal{M}_A \equiv \mathcal{F}. \end{aligned}$$

Thus, the form of the hybrid fault model that applies to a given node or node set is determined by the worst-case faults that are assumed or shown to occur.[3]

$H_{\mathcal{B}}$: The worst-case faults can be shown or assumed to be non-malicious, and the fault set is $\mathcal{F}_{\mathcal{B}}$. Hybrid active redundancy algorithms can be used (See §2.4.), with $m_{\mathcal{B}}$ processes needed to tolerate $f_{\mathcal{B}}$ faults, where $m_{\mathcal{B}} \geq f_{\mathcal{B}} + 1$.

$H_{\mathcal{MS}}$: The worst-case faults can be shown or assumed to be malicious symmetric, and the fault set is $\mathcal{F}_{\mathcal{MS}}$. Hybrid non-iterative passive redundancy algorithms, which handle mixed faults must be used. (See §2.5.) A total of $m_{\mathcal{MS}}$ processes is needed to tolerate $f_{\mathcal{MS}} = f_{\mathcal{B}} + 2f_{\mathcal{M}_s}$ faults, with $m_{\mathcal{MS}} \geq f_{\mathcal{MS}} + 1$.

$H_{\mathcal{MA}}$: The worst-case faults can be shown or assumed to be malicious asymmetric. The fault set is $\mathcal{F}_{\mathcal{MA}} \equiv \mathcal{F}$, which means that all possible faults are addressed. An iterative passive redundancy algorithm is required, with the algorithm $Z(r)$ of the unified model [11] sufficient for interactive consistency. If the $Z(r)$ algorithm is used, then $m_{\mathcal{MA}}$ nodes can tolerate a total of $f_{\mathcal{MA}} = 2f_{\mathcal{M}_A} + 2f_{\mathcal{M}_s} + f_{\mathcal{B}}$ faults, with $m_{\mathcal{MA}} \geq f_{\mathcal{MA}} + r + 1$.
For other hybrid iterative passive redundancy algorithms, a minimum of $m_{\mathcal{MA}}$ nodes is sufficient to tolerate $f_{\mathcal{MA}} = 3f_{\mathcal{M}_A} + 2f_{\mathcal{M}_s} + f_{\mathcal{B}}$ faults, with $m_{\mathcal{MA}} \geq f_{\mathcal{MA}} + 1$. Such hybrid interactive consistency and convergence algorithms will need to be based on the hybrid fault-tolerant voting functions presented in §2.5.

[3] It may also be sufficient to demonstrate extremely low probabilities of occurrence for faults excluded from the assumed fault set.

In future work, we will provide a reliability model for the hybrid fault model, compare it with the static reliability models presented in §2.2, and address the issue of imperfect fault coverage.

2.4 Hybrid Active Redundancy Techniques

In the active redundancy techniques developed to support the hybrid fault model, we examine each message or value received by a node using an active redundancy technique. If a non-malicious fault is detected, such as a framing, parity, or encoding fault, a missing message, or a range violation, then we adopt a default error or status value, $v_{\mathcal{B}}$, as the value received by the node in the message. Under no circumstances can $v_{\mathcal{B}}$ be an acceptable value, and it may differ based on the data types of correct values. Without loss of generality, we assume that the value $v_{\mathcal{B}}$ is greater than any permitted numerical data value. We also assume perfect detection, as faulty nodes are not of concern. The only difference between hybrid and standard active redundancy techniques is in the specific action taken to adopt a default error value, $v_{\mathcal{B}}$, when a non-malicious fault is detected.

2.5 Hybrid Fault-Tolerant Voting Functions

Passive redundancy techniques usually employ voting functions designed to tolerate a predefined number of faults based on the total number of values they receive. We extended these functions to accomodate the default error or status value, $v_{\mathcal{B}}$, which is the consequence of a non-malicious fault under the hybrid active redundancy techniques defined above.

We define the function *exclude*$(V) = V_{\mathcal{E}}$, which takes a set of m elements, $V = \{v_1, v_2, \ldots, v_m\}$, removes any error values, $v_{\mathcal{B}}$, from V, and returns the set $V_{\mathcal{E}}$, containing $m_{\mathcal{E}}$ elements. The set $\mathcal{E}$ is the original fault set $\mathcal{F}$ with the $f_{\mathcal{B}}$ non-malicious faults in $\mathcal{F}_{\mathcal{B}}$ removed, and $m_{\mathcal{E}} = m - f_{\mathcal{B}}$. In the absence of non-malicious faults ($f_{\mathcal{B}} = 0$), $V \equiv V_{\mathcal{E}}$, as no elements are excluded. Hybrid voting functions are based on the *exclude()* function.

2.5.1 Hybrid Majority Vote

A majority vote is typically used by each good node to compute a common final value for bimodal values received from other nodes or input sources. Let V be a set of m elements, as described above, with v_0 a default value, which could potentially be taken by any v_i. Then, we have

$$\mathit{majority}(\mathit{exclude}(V)) = \begin{cases} v, & \text{if more than } \lfloor \frac{m_{\mathcal{E}}-1}{2} \rfloor \text{ of the } v_i = v. \\ v_0, & \text{otherwise.} \end{cases}$$

The default value, v_0, returned when no majority exists must be defined *a priori* and must be a potentially correct value, to avoid introducing a fault into a fault-free scenario. Since the *majority* function ignores $\lfloor \frac{m_{\mathcal{E}}-1}{2} \rfloor$ elements, the composite function tolerates up to f faults, where $f = f_{\mathcal{B}} + \lfloor \frac{m_{\mathcal{E}}-1}{2} \rfloor$.

2.5.2 Hybrid Mean and Midpoint

The functions *mean* and *midpoint* are commonly used to average numerical data. However, in their raw state they are sensitive to extreme values. So, we defined fault-tolerant versions of these functions using the *reduce* function, where, if V is a set of m values to be voted, then:

$$\mathit{reduce}(V, t) = \{V\} - \{\text{the } t \text{ largest and } t \text{ smallest } v_i\}.$$

To tolerate the most faults, t is taken to be the maximum number of faults tolerable by m elements, which is $t(m) = \lfloor \frac{m-1}{2} \rfloor$.

The *mean* of m values $v_i \in V$, for $i = 1, \ldots, m$, is $mean(V) = \frac{1}{m}\sum_{i=1}^{m} v_i$.
The *midpoint* of m values $v_i \in V$, for $i = 1, \ldots, m$, is the mean of extrema, with $midpoint(V) = \frac{1}{2}(\min_{i=1,m}(v_i) + \max_{i=1,m}(v_i))$, often called the *mean of medial extremes* (MME).

The *hybrid fault-tolerant mean* and *hybrid fault-tolerant MME* functions are achieved by applying the *mean* and *midpoint* functions to restricted subsets of values, where the restriction first removes the values $v_{\mathcal{B}}$ (from detected non-malicious faults) then eliminates the extrema from the remaining elements using the *reduce* function. The number of extrema eliminated now depends on $m_{\mathcal{E}}$, the number of elements remaining after removing the $f_{\mathcal{B}}$ non-malicious fault values $v_{\mathcal{B}}$. So, we have

$$hybrid\text{-}fault\text{-}tolerant\ mean(V) = mean(reduce(exclude(V), t(m_{\mathcal{E}})))$$
$$hybrid\text{-}fault\text{-}tolerant\ MME = midpoint(reduce(exclude(V), t(m_{\mathcal{E}}))),$$

with $t(m_{\mathcal{E}}) = \lfloor \frac{m_{\mathcal{E}}-1}{2} \rfloor$. Each function tolerates a total of $f = f_{\mathcal{B}} + \lfloor \frac{m_{\mathcal{E}}-1}{2} \rfloor$ faulty elements.

2.5.3 Hybrid Median

The *median* or *median-select* voting function returns the middle value of an odd number of ordered elements, or the average of the two middle values of an even number of ordered elements. Since extreme values are ignored, the *median* is inherently fault-tolerant. So, the *hybrid-median* consists of the *median()* applied after the *exclude()* function. For a set V of m ordered values $\{v_1, v_2, \ldots, v_m\}$, where $v_q \leq v_{q+1}$,

$$hybrid\text{-}median(V) = \quad median(exclude(V)) = \quad \frac{(v_i + v_j)}{2},$$

where $i = 1 + k$, $j = m_{\mathcal{E}} - k$, and $k = \lfloor \frac{m_{\mathcal{E}}-1}{2} \rfloor$. Since $v_q \leq v_{\mathcal{B}}$ by definition, the excluded values $v_{\mathcal{B}}$ will be the $f_{\mathcal{B}}$ largest values. So, the elements remaining in $V_{\mathcal{E}}$ after application of the *exclude* function will be $\{v_1, v_2, \ldots, v_{m_{\mathcal{E}}}\}$.

In future work, we will develop the interactive convergence and interactive consistency techniques (using algorithms besides $Z(r)$) which take advantage of the mixed fault types of the hybrid fault model. The basis for these algorithms will be the hybrid fault-tolerant voting functions presented in this section.

2.6 Fault Classification and Containment

Since the fault sets $\mathcal{B}_S$, $\mathcal{B}_A$, $\mathcal{M}_S$, and $\mathcal{M}_A$, are mutually exclusive and collectively exhaustive, we used the hybrid fault model as the basis for a simple fault classification and system analysis algorithm, the *Hybrid Fault Analysis Algorithm.* We also derived directions for future research by applying this algorithm to a few simple examples.

2.6.1 Hybrid Fault Analysis Algorithm (HFA)

Assume there is a single transmitting node, T, sending messages to k receiving nodes, R_i, for $i = 0, 1, \ldots, k$, in a set of m nodes. Without loss of generality, assume that a single output message is generated by T, with the true value given by v, and that node R_i receives value e_i.[4] If no message is received by a receiver R_j, then $e_j = \emptyset$.

The basic philosophy in discerning potential fault malice is to determine the ***worst-case*** errors produced in messages by the faulty node and then to decide what type of active redundancy techniques, if any, can be used to detect the faulty node. If such a technique is implemented in the receiving node, then the fault is ***non-malicious***. Otherwise, it is ***malicious***. With the malice of the fault determined, an examination of the worst-case error values indicates fault symmetry. We consider the output sets corresponding to the worst case errors, represented by $\{e_1, \ldots, e_k\}$. If $v \neq e_i$ and $e_i = e_j$ for all i and j, for all such errors, then the worst-case node fault is ***symmetric***; otherwise, it is ***asymmetric***. Once the type of the worst-case fault has been determined, the hybrid fault model indicates the type of algorithm required to tolerate the fault in that output, which can be compared to the algorithms actually implemented.

2.6.2 HFA Evaluation Results

The fault scenarios below indicate the the uses of the HFA, as well as future work needed to make the algorithm more robust.

Suppose the worst-case fault were determined to be malicious symmetric. Then, by $H_{\mathcal{MS}}$, a passive redundancy technique must be implemented in the receivers to tolerate that fault. If the system uses only active redundancy techniques, then the probability of the worst-case fault should be demonstrated to be extremely low, or a malicious symmetric fault could represent a single point of failure.

The worst case faults are not the only consideration of the HFA, as shown by the following example. Suppose that a fault f from a sender is classified as malicious because the range check that could detect f is not implemented in the receiver. If the receiving node is redesigned to implement a range check, or any other technique which will detect f, then the malicious fault f is transformed into a non-malicious fault. Thus, the potential for fault transformation is indicated using the HFA.

It is also possible for several faults to be active in T, producing errors which are seen as a single fault by each non-faulty node R_i, or which are *never* seen by any non-faulty node R_i, as described in the following example. Suppose that T is faulty, *i.e.*, there exists at least one fault f in T. Consider all possible output sets $\{e_1, \ldots, e_k\}$ which the node T could produce in the presence of f. If, for all such output sets, $v = e_i$ for all i, then no error will be produced by the faulty transmitting node. This does not mean that T is not faulty, only that the fault f has been *contained* within node T, and the non-faulty R_i cannot discern the fault in T because no errors are manifested in the messages they receive. While precise fault classification improves reliability estimates by indicating probable and improbable errors, this example shows that the manner in which faults are counted is just as important as their classification. We will address these issues in future work on fault containment.

The HFA does not address the effects of the faulty node T on different sets of receiving nodes simultaneously, nor the effects of several faulty nodes. Furthermore, at the receiver level, an error in a message from T is viewed as a *single node fault,* since the one node transmitting is assumed to be faulty, and not the communication link.

[4] We use a single output in this algorithm to represent the fact that all nodes R_i receive the same set of information from T in a fault free scenario. In practice, the single output is a message or a set of messages.

By combining the HFA algorithm with the error manifestation taxonomy, presented in [1], we can develop a more robust algorithm that deals with these issues. Our future work also will address fault transformation and will evaluate and construct fault containment regions, *i.e.*, system partitions, components or nodes with independent failure probabilities, that limit the scope of faults.

References

[1] M. C. McElvany and C. J. Walter, "ONR contract N00014-91-C-0014, quarterly report," quarterly report, 91-Jan-01 to 91-Apr-31, ATC, Allied Signal Aerospace Company, Columbia, MD 21045, May 1991.

[2] B. W. Johnson, *Design and Analysis of Fault-Tolerant Digital Systems.* Addison-Wesley Publishing Company, 1989.

[3] D. Dolev, N. Lynch, S. Pinter, E. Stark, and W. Weihl, "Reaching approximate agreement in the presence of faults," in *Proceedings, Third Symposium on Reliability in Distributed Systems*, pp. 145–154, October 1983.

[4] L. Lamport and P. Melliar-Smith, "Byzantine clock synchronization," in *Proceedings, Third ACM Symposium on Principles of Distributed Computing*, pp. 68–84, August 1984.

[5] D. Dolev *et al.*, "An efficient algorithm for Byzantine agreement without authentication," *Journal of Information and Control*, vol. 52, pp. 257–274, 1982.

[6] F. Cristian, H. Aghili, and R. Strong, "Atomic broadcast: From simple message diffusion to Byzantine agreement," in *Proceedings, Fifteenth International Symposium on Fault Tolerant Computing*, pp. 200–206, IEEE, June 1985.

[7] F. Meyer and D. Pradhan, "Consensus with dual failure modes," in *Proceedings, Seventeenth International Symposium on Fault Tolerant Computing*, pp. 48–54, IEEE, July 1987.

[8] O. Babaoglu and R. Drummond, "Streets of Byzantium: Network architectures for fast reliable broadcasts," *IEEE Transactions on Software Engineering*, vol. SE-11, pp. 546–554, June 1985.

[9] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *JACM*, vol. 27, pp. 228–234, April 1980.

[10] B. S. Dhillon, *Reliability in Computer System Design.* Norwood, NJ 07648: Ablex Publishing Corp, 1987.

[11] P. Thambidurai and Y.-K. Park, "Interactive consistency with multiple failure modes," in *Proceedings, Seventh Symposium on Reliable Distributed Systems.* pp. 93–100, IEEE, October 1988.

[12] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, pp. 382–401, July 1982.

[13] P. Thambidurai, Y.-K. Park, and K. Trivedi, "On reliability modeling of fault-tolerant distributed systems," in *Proceedings, 9th International Conference on Distributed Computing Systems*, IEEE, June 1989.

Principal Investigator(s) Name: C. J. Walter, M. C. McElvany
PI Institution: Aerospace Technology Center, Allied-Signal Aerospace Company
PI Phone Number: (301) 964-4082, (301) 964-4158
PI E-mail Address: chris@batc.allied.com, michelle@batc.allied.com
Contract Title: Reliability Modeling of Dependable Distributed Systems
Contract Number: N00014-91-C-0014
Reporting Period: 91–May–01 to 91–Jul–31

3 Research Transitions

We are transferring results to our internal IR&D projects on a continuing basis.